Exhibit B

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

EMC CORPORATION,

Petitioner

v.

ACQIS LLC,

Patent Owner

Case No. IPR2014-01462
Patent: 8,041,873

PATENT OWNER'S RESPONSE

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

LIST OF EXHIBITS

Exhibit 2001	*PCI Local Bus Specification*, Production Version, Rev. 2.1, June 1, 1995 ("PCI Local Bus Specification")
Exhibit 2002	*Computer Desktop Encyclopedia*, (2d Ed. 1999) ("Computer Desktop Encyclopedia")
Exhibit 2003	*Microsoft Press Computer Dictionary*, (3d Ed. 1997).
Exhibit 2004	CERN DOCUMENT SERVER, Listing for Bogaerts, A. et al., *RD24status report: application of the scalable coherent interface to data acquisition at LHC*, http://cds.cern.ch/record/294145/files/.
Exhibit 2005	CERN LIBRARY RESOURCES, http://library.web.cern.ch/resources.
Exhibit 2006	P.R. 4-3 Joint Claim Construction Statement, *ACQIS v. EMC Corp.*, 6:13-cv-00638-LED (E.D. Tex. Nov. 25, 2014)
Exhibit 2007	Horst, TNet: A Reliable System Area Network, 15 IEEE Micro 37 (Feb. 1995) ("Horst") as annotated by Bruce Young.
Exhibit 2008	Horst, TNet: A Reliable System Area Network, 15 IEEE Micro 37 (Feb. 1995) ("Horst") as annotated by Bruce Young.
Exhibit 2009	Bogaerts, Application of the Scalable Coherent Interface to Data Acquisition at LHC (Oct. 1996) as annotated by Bruce Young.
Exhibit 2010	Bogaerts, Application of the Scalable Coherent Interface to Data Acquisition at LHC (Oct. 1996) as annotated by Bruce Young.
Exhibit 2011	Declaration of Spencer Scott dated April 8, 2015.
Exhibit 2012	Computer Architecture, A Quantitative Approach, J. Hennesy & D. Patterson, 1990 (Excerpts)
Exhibit 2013	Computer Organization and Design, the Hardware/Software Interface, D. Patterson & J. Hennessy, 1998 (excerpts)
Exhibit 2014	PCI Express System Architecture, by R. Budruk et al, 2004 (excerpts)
Exhibit 2015	PCI and PCI-X Hardware and Software Architecture & Design, E. Solari & G. Willse, 2001 (excerpts)
Exhibit 2016	Intel Nontransparent Bridge Spec., 21554 PCI-to-PCI Bridge, Dec. 1996
Exhibit 2017	VITA Journal, Multiprocessing on PCI and CompactPCI, "Here Today – Multiprocssing on PCI and CompactPCI", October/November/December 1998
Exhibit 2018	US Patent No. 8,041,873, October 18, 2011
Exhibit 2019	PCI Express Base Specification, Revision 3.0, November 10, 2010
Exhibit 2020	RapidIO Technology Comparisons _ PCIe and Ethernet vs. RapidIO

LIST OF EXHIBITS
(continued)

Exhibit 2021	Declaration of Volker Lindenstruth, dated June 11, 2015
Exhibit 2022	Deposition of Bruce Young, dated May 20, 2015, Case No. IPR2015-01462 & 2014-01469
Exhibit 2023	Deposition of Bruce Young, dated May 21, 2015, Case No. IPR2015-01462 & 2014-01469
Exhibit 2024	IBM Dictionary of Computing, G. McDaniel, 1994 (excerpts)
Exhibit 2025	Deposition of Spencer Scott, dated June 1, 2015, Case No. IPR2015-01462 & 2014-01469
Exhibit 2026	Declaration of Christopher Butler, dated March 30, 2015

I. INTRODUCTION

At first glance, there is superficial similarity between the primary reference presented to the Board and the '873 patent claims at issue here. But that similarity belies critical, patentable distinctions between the '873 inventions and the Horst reference. As explained below and confirmed by Dr. Lindenstruth, even though a PCI transaction can eventually be completed in Horst, these transactions occur on a remote, *parallel* PCI bus and are never serialized, as required by the claims at issue. Ex. 2021 ¶¶ 63, 119, 136, 144. This distinction, and others, differentiates the '873 patent claims from the prior art in the instituted grounds.

The primary reference applied by the Board solves a very different problem than the '873 inventions. Horst is designed for massively-parallel systems in which one CPU can share the memory space of hundreds of other CPU nodes. Ex. 1011 at 1. Sharing CPU memory space between multiple CPUs is not supported by the Peripheral Component Interconnect ("PCI") local bus standard because of address collisions and overlapping address spaces. Ex. 2104 at 6-7; Ex. 2016 at 6; Ex. 2017 at 3-4; Ex. 2020 at 3.

Each CPU in the TNet system has the same processor address space. Ex. 1011 at 5; Ex. 2021 ¶¶ 139, 152. The PCI standard's addressing scheme is only designed to work with a single host processor's address space. Ex. 2016 at 6; Ex.

2017 at 3; Ex. 2020 at 3. *Id.* A read-or-write PCI transaction using a PCI address would cause a collision because every CPU would have the same address. Ex. 2016 at 6; Ex. 2017 at 3-4; Ex. 2020 at 3; Ex. 2021 ¶ 79. To solve this addressing problem, Horst discarded the PCI standard bus protocol and introduced a new protocol—TNet. TNet permits one CPU to directly address the CPU memory of another CPU. Ex. 2021 ¶¶ 102-104. TNet also permits generation of a PCI transaction at a remote device. Ex. 1011, Fig. 2. But notably, TNet eliminates the PCI transaction on the CPU side of the network. Eliminating the PCI transaction on the CPU side was necessary to enable the massively-parallel system. Because PCI was in widespread use for peripheral devices, Horst needed to provide access to these devices. Ex. 2021 ¶ 84. To maintain PCI compatibility, Horst implemented hardware and software interfaces *on the peripheral side* designed to generate a PCI transaction for transmission only over the remote, parallel PCI bus. *Id.* ¶¶ 102-104 Horst does not employ PCI transactions within its own TNet fabric, and TNet never puts a PCI transaction on a serial bus. *Id.* ¶¶ 63, 119, 136, 144.

The '873 invention solves a very different problem than Horst was addressing. The '873 invention is directed at increasing communication speeds between the computer and peripherals. *Id.* ¶¶ 84. The inventor recognized that, as CPU speeds increased, communications with peripheral devices would need to

keep up. *Id.* But the problem with keeping up was that a limiting industry standard, PCI, had been widely adopted and could not be replaced without requiring peripheral manufacturers to replace drivers (the software that controls the peripheral connection). *Id.* ¶¶ 84, 89.

Realizing that the industry had invested heavily in PCI and would not easily abandon it, even if it was too slow, the '873 inventor developed a system to speed up PCI transactions that was completely compatible with existing peripheral devices and drivers. *Id.* ¶¶ 90, 130. As described in the specification and explained further below, one key to the invention was to serialize the otherwise parallel PCI bus transactions to increase communication speeds for peripherals. *Id.* ¶¶ 85-90. Another key was to maintain the PCI transaction address bits as generated on the CPU side of the network so that the serialized communications were compatible with existing peripheral devices and their drivers. Ex. 2020 at 4:50-58; *Id.* ¶¶ 75, 85-90. The claims reflect this invention with their focus on the serialized PCI bus transaction, the serialized address bits of the PCI bus transaction, and the specific architecture in which the serialized PCI bus transaction is transmitted from the computer to the peripheral. The inventor chose claim terms like "directly coupled," "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction," and "encoded PCI address and

data bits" to distinguish the claimed architecture from other systems. Ex. 1001, claims 54 & 61.

Because Horst solves a very different problem, it is not surprising that it only matches up superficially with the '873 claims. But, as the arguments below describe in detail, when the specific language of the claim limitations are considered, such as "directly coupled," "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction," and "encoded PCI address and data bits," Horst fails to disclose the claimed invention, and the secondary references do not cure the failure. Specifically, to implement its parallel-processing systems with shared CPU memories, Horst deliberately eliminated the very principle that the '873 claims address—a CPU-side PCI bus transaction that increases communication speed. And Horst never used a serialized PCI bus transaction, only parallel. Ex. 2021 ¶¶ 102-104.

Figure A[1] illustrates the difference between Horst and the traditional PCI-based system. Figure 1-2 from the PCI specification is shown on the left. Ex. 2001 at Fig. 1-2. Figure 2 of Horst is shown on the right. Ex. 1011 at Fig. 2. The traditional system included a CPU side (labeled as "A") that communicated with a

[1] To avoid confusion with the numbered figures pulled from the cited references, Response figures are lettered (e.g., Figure A).

peripheral side (labeled as "B") through a parallel PCI local bus (labeled as "C"). Horst removed the PCI bus and replaced it with a TNet network and a proprietary protocol. The TNet system (shown on the right) has a CPU side (labeled as "A_1") that communicates with a peripheral side (labeled as "B_1") through the TNet link (labeled as "C_1"). Figure A illustrates that, in the upper portion of Horst, which is the CPU side, there is no PCI bus. And, consequently, the TNet link does not generate PCI bus transactions or PCI transactions at the CPU side.

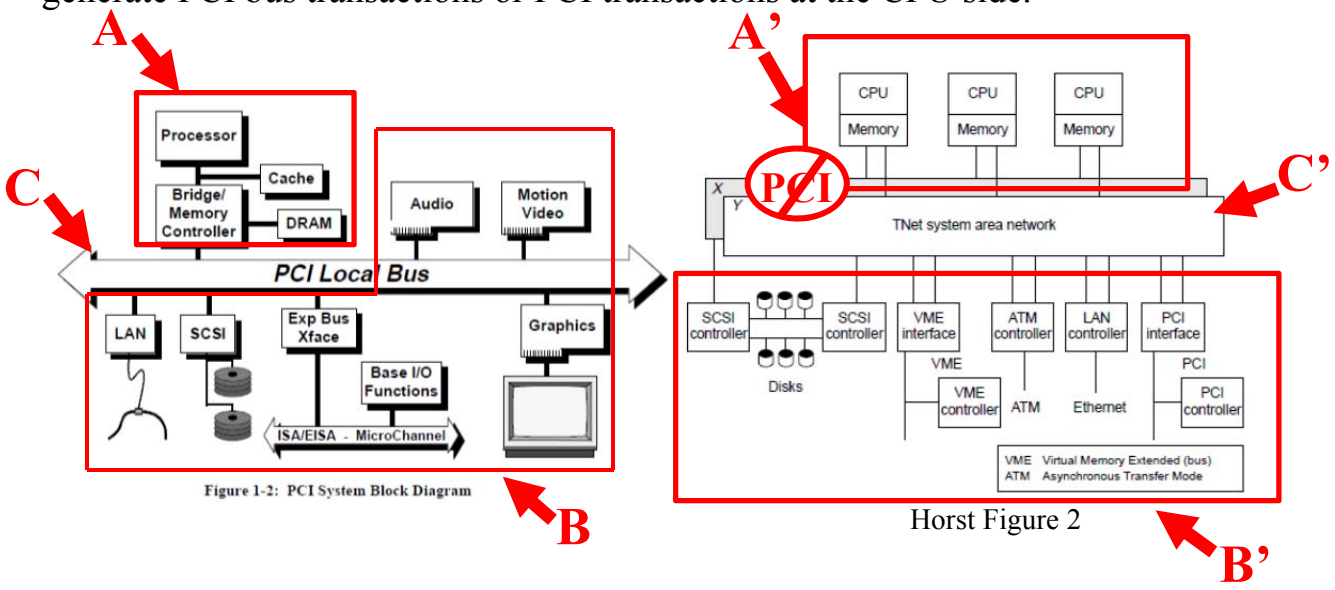


Figure A: Comparison of the traditional PCI-based system and TNet

Just because a PCI transaction is eventually completed elsewhere in Horst on a remote PCI bus—on the peripheral side—does not make Horst the same as the '873 invention. The '873 patent claims a very specific architecture linked to serializing a PCI bus transaction on the CPU side of a network as opposed to the peripheral side. Ex. 1001, claim 54. Horst did not and could not implement this

CPU-side PCI serialization and still achieve its massively-parallel processor goals. Ex. 2021 ¶¶ 55-56, 78, 102-103. Accordingly, Horst does not teach the limitations recited in the claims under review. And, no other references fill, or are even offered to fill, these gaps.

II. BACKGROUND AND SUMMARY OF ARGUMENTS

A. Related Matters

IPR2014-01469 has been instituted on related U.S. Patent No. RE42,814.

The '873 patent has been asserted in two pending district court litigations. *ACQIS LLC v. Alcatel-Lucent USA Inc*., 6:13-cv-638, pending before the United States District Court for the Eastern District of Texas and *ACQIS, LLC v. EMC Corporation,* 1:14-cv-13560, pending and stayed before the United States District Court for the District of Massachusetts.

B. Grounds Upon Which Trial is Instituted

Ground 1: Claims 54, 57, 60, and 61 are obvious over Horst (Ex. 1011) and the LVDS Owner's Manual (Ex. 1019);

Ground 2: Claims 56 and 59 are obvious over Horst (Ex. 1011), the LVDS Owner's Manual (Ex. 1019), and Pocrass (Ex. 1016);

Ground 3: Claim 58 is obvious over Horst (Ex. 1011), the LVDS Owner's Manual (Ex. 1019), and Deters (Ex. 1017).

C. Background of the Invention of the '873 Patent

The '873 patent describes many features that could be used in modularized computer systems. At its most basic level, the patent describes a modular computing system that provides backup capability, dual processing, and the like. Ex. 1001 at 4:10-16. But the patent goes much deeper into different modular architectures that are (1) compatible with existing communication protocols and (2) designed for increasing communication speed with peripherals in a modular computer system. *Id.* at 4:50-58. The '873 claims are directed at an industry-compatible architecture that speeds up communications between the attached computer module ("ACM") and peripheral devices. *Id.* at claim 54. More specifically, the '873 claims are directed to a CPU-side (as opposed to a peripheral-side) architecture that permits faster PCI bus transactions with peripheral devices by serializing the PCI bus transaction, including serializing the PCI transaction address. Ex. 2021 ¶¶ 31-35, 85-90.

One significant advantage to the use of PCI as the communication between the CPU side and the peripheral side is the speed of communication with peripheral devices. *Id.* ¶¶ 61-62. The invention of the '873 patent retains that advantage, as well as compatibility with existing devices and PCI drivers, while boosting speeds further by making the PCI communication serial rather than parallel.

Note that PCI is an industry standard. *Id.* ¶ 60; Ex. 2001. Standards are very important in computer architecture. Standards ensure that multiple designers can develop hardware and software and that it will all work together. Even minor deviations from a standard will result in incompatible hardware and software. *Id.* ¶ 65. Therefore, adherence to the specification of a standard (such as the PCI standard local bus specification) is critical for interoperability with other components of a system. The '873 patent adheres to the PCI standard. TNet does not. Ex. 1011 at 2.

1. The PCI Standard

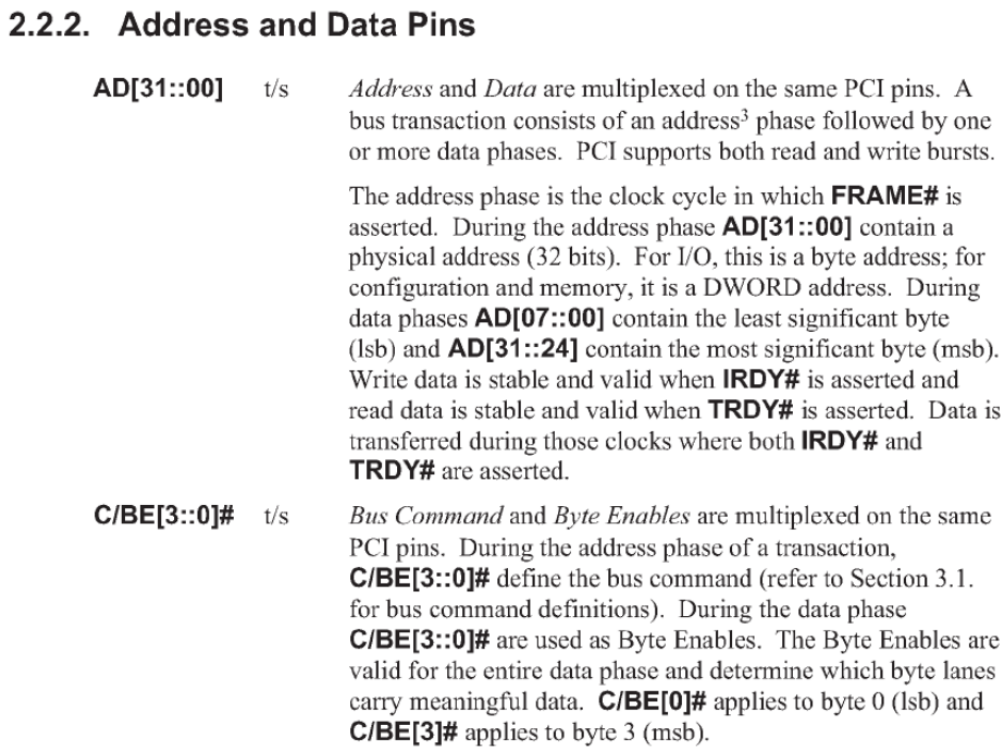
In the late 1990s, the PCI bus was ubiquitous in computers and provided interconnection between processors and peripheral devices. Ex. 2021 at ¶ 60. This bus operated according to the PCI standard, which was developed by Intel and maintained by an industry group, the PCI Special Interest Group ("PCI-SIG"). *Id.* ¶¶ 60, 91. Peripheral manufacturers were expected to provide drivers for their devices so that they could interact with the PCI bus found on nearly every computer. *Id.* And the PCI standard provided the instructions for designing those drivers. *Id.* ¶¶ 60-83.

The PCI bus was a 32-bit wide parallel bus with multiplexed address and data bits and separate command and timing control lines. Ex. 2001 at 17; Ex. 2021 ¶ 63. According to the published PCI standard, a PCI bus transaction consists of

an address phase followed by one or more data phases. Ex. 2001 at 25; *Id.* ¶ 64. Each device on a PCI bus must have a unique physical address. Ex. 2021 ¶ 65. And any device on the PCI bus can read or write data directly to or from another device's address. *Id*.

The PCI specification defines specific bus transactions that use the three different PCI physical address spaces, including: memory read-and-write transactions, I/O read-and-write transactions, and configuration read-and-write transactions. Ex. 2001 at 25. These transaction types are defined by bus commands that tell the PCI devices which address space is being utilized in the particular transaction. Ex. 2001 at 21, 37; Ex. 2021 ¶ 73. Specifically, the PCI transaction types define different address spaces and different address formats for addressing PCI devices on a PCI bus. Ex. 2001 at 41; Ex. 2021 ¶ 73.

A PCI standard bus transaction requires the PCI bus command specific to the transaction type and a corresponding physical PCI address in the correct format and to the correct address space. Ex. 2021 ¶ 76. The PCI bus transaction must comply with a particular format for peripheral devices and their PCI drivers to understand it. Figure B illustrates the specific requirements of PCI address and data pins from the PCI standard local bus specification. Ex. 2001 at 25.



2.2.2. Address and Data Pins

AD[31::00] t/s *Address* and *Data* are multiplexed on the same PCI pins. A bus transaction consists of an address[3] phase followed by one or more data phases. PCI supports both read and write bursts.

The address phase is the clock cycle in which **FRAME#** is asserted. During the address phase **AD[31::00]** contain a physical address (32 bits). For I/O, this is a byte address; for configuration and memory, it is a DWORD address. During data phases **AD[07::00]** contain the least significant byte (lsb) and **AD[31::24]** contain the most significant byte (msb). Write data is stable and valid when **IRDY#** is asserted and read data is stable and valid when **TRDY#** is asserted. Data is transferred during those clocks where both **IRDY#** and **TRDY#** are asserted.

C/BE[3::0]# t/s *Bus Command* and *Byte Enables* are multiplexed on the same PCI pins. During the address phase of a transaction, **C/BE[3::0]#** define the bus command (refer to Section 3.1. for bus command definitions). During the data phase **C/BE[3::0]#** are used as Byte Enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. **C/BE[0]#** applies to byte 0 (lsb) and **C/BE[3]#** applies to byte 3 (msb).

Figure B: PCI Local Bus Standard Specification – Address and Data Pins

2. The '873 patent claimed invention increases communication speed while maintaining the CPU-side PCI bus transaction

PCI was the industry standard in the late 1990s. Executing a PCI bus transaction was a requirement for any new, commercially-acceptable computer system. But PCI had a major problem—speed constraints. The PCI standard specified 32-bit parallel communications. Ex. 2021 ¶ 63. The inventor of the '873 patent recognized that parallel communications would eventually become a speed bottleneck for communicating with peripheral devices. *Id.* ¶¶ 34-35.

The inventor recognized that a serial communication scheme would allow fast, reliable communication. *Id.* 34-35, 85. Serial communication is faster than

parallel communication because parallel communications are slowed by data skew and electromagnetic interference between closely-positioned parallel lines (known as cross-talk). *Id.* ¶ 45, 59. But designing a new, serial communication protocol was not practical. The trick was to use the existing PCI standard and make it faster for communicating with peripheral devices. And the solution is in the '873 claims—a new architecture that serializes the PCI bus transaction from the CPU side of the network (rather than using an entirely new standard and moving the old parallel PCI bus to the peripheral side). By retaining PCI communication on the CPU side, (including, for example, address, command, and configuration), the system does not require new drivers or peripheral devices. Ex. 2020 at 4:40-58; Ex. 2021 ¶ 84, 89. Further, the claimed invention does not suffer performance penalties because the serialized PCI connection is as fast as the PCI standard and no additional address translations or software intervention is required. Ex. 2020 at 4:50-58; Ex. 2021 ¶ 85.

By using the industry standard PCI bus transaction, this new, claimed architecture not only increased communication speeds, but it was also compatible with existing peripherals and their drivers. *Id.* Further, by maintaining the PCI transaction, the new architecture also took advantage of the plug-and-play capabilities of the PCI standard. *Id* ¶ 89. Accordingly, the '873 patent describes—

and claims—the solution to the problem caused by the PCI standard's use of parallel communications while maintaining compatibility with that standard.

3. The '873 patent claims specifically claim the improved architecture that maintains the CPU-side PCI bus transaction

The claims specifically address the architecture required to implement PCI bus transactions on the CPU side and to achieve the speed gains.

Referring first to the speed gains, the claims address the speed gains with the use of the low voltage differential signal ("LVDS") channel. Ex. 1001, claim 54; Ex. 2021 ¶ 90. The channel communicates data serially. *Id.* The claim language in claim 54 makes clear that the channel communicated data as a serial bit stream. For example, claim 54 refers to "two sets of unidirectional, multiple serial channels" and "encoded serial bit stream." Ex. 1001, claim 54. This serial bit stream of a PCI bus transaction increases speed over parallel forms of a PCI bus transaction. Ex. 2021 ¶ 59, 85.

Referring now to the CPU-side architecture, the claims define a specific architecture that distinguishes it from the peripheral-side architecture of Horst. For example, claim 54 recites a peripheral bridge "directly coupled" to the processing unit. Ex. 1001, claim 54. This "directly coupled" limitation establishes the peripheral bridge as part of the CPU side and not the peripheral side of the system.

Claim 54 specifically requires that the peripheral bridge communicate the serialized PCI bus transaction data bits over a low-voltage differential signal ("LVDS") serial channel that is directly coupled to the peripheral bridge. Accordingly, the PCI bus transaction exists on the CPU side of the system as opposed to only on the peripheral side.

D. Background of Horst

Horst did not address the communication speeds between the computer and its peripheral devices. It was directed at a protocol for massively-parallel systems that needed to share CPU memory among multiple processors. While TNet enables communications between a computer and a peripheral device, it does so by eliminating the CPU-side PCI transaction and replacing it with a brand-new communication protocol. TNet requires that a PCI transaction be created on the peripheral side of the network by a custom piece of hardware and software. Simply put, Horst does not solve the peripheral communication-speed problem because it was not trying to.

1. Horst did not solve the communication-speed problem and does not use a CPU-side PCI bus transaction

Horst (TNet) interconnected hundreds of processors to share memory among the individual CPUs while allowing the processors to share I/O resources. Ex. 1011 at 1. The author evaluated existing communications standards, which

included PCI, but "found nothing that met all the requirements." *Id.* at 2. "This forced [the author] into the difficult job of designing a completely new network." *Id.* at 2. As a result, the author specifically chose to avoid the PCI standard bus transaction protocol on the CPU side as well as any other existing protocol in favor of a "completely new network." *See Id.* at 2, 5, Fig. 5.

The "completely new network" described by Horst requires specially designed intelligent processor and I/O nodes driven by application-specific integrated circuits (ASICs) to generate and transmit TNet transactions. Ex. 1011 at 6-7, Figs. 7, 8. The "completely new network" described by Horst uses virtual (rather than physical) addressing and address translation and purposefully avoids memory-mapped I/O and physical addressing (e.g., PCI). *Id.* at 2, 6. Stated differently, Horst needed to introduce new peripheral-side hardware and software to enable a PCI transaction because his new protocol eliminated PCI transactions from the CPU side. All communications on the CPU side are done exclusively in the proprietary TNet protocol. The result is a system riddled with proprietary hardware and software interfaces to facilitate communication between the CPU and legacy devices. *Id.* ¶ 116. Figure C, excerpted and annotated from Horst figure 2, illustrates the CPU side and peripheral side of the TNet system. Note that the PCI communication exists *only* in the peripheral side.

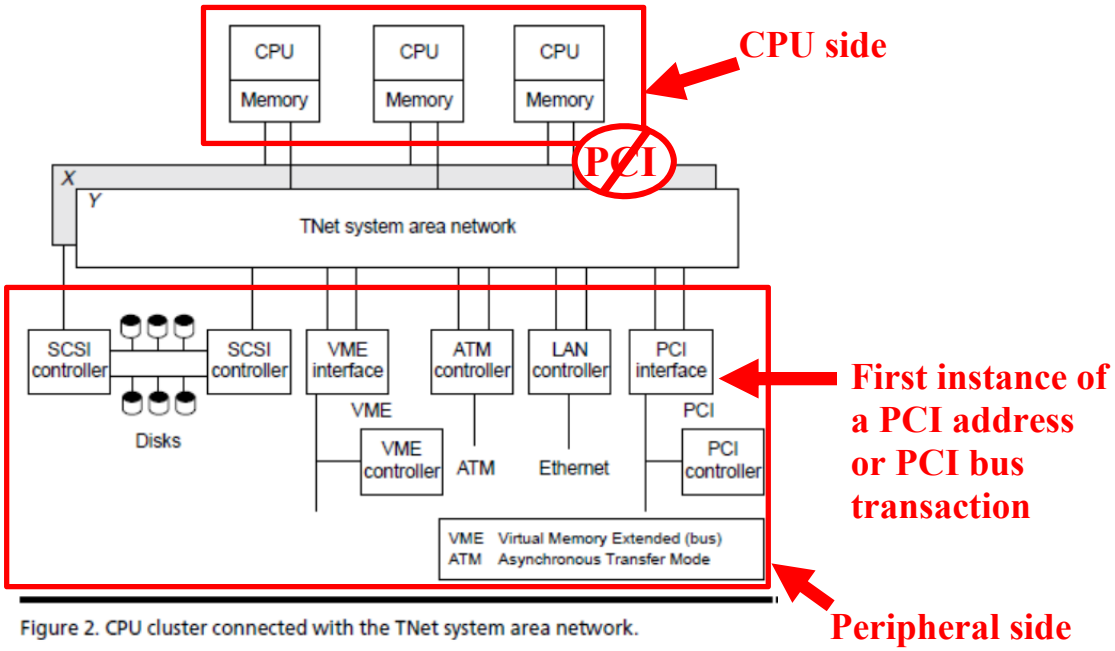


Figure C: Annotated TNet

While there is a lack of detail around the exact routing mechanisms and address schemes used on the TNet protocol, as Dr. Lindenstruth explains, it is clear from the TNet reference that there is no PCI standard bus transaction on the TNet link. Ex. 2021 at ¶¶ 114, 134-146. The disclosed TNet transaction types are only read, write, and unacknowledged write. Ex. 1011 at 5, Fig. 5. The TNet packets and transaction types do not allow for the PCI standard bus transactions, including memory read and write, I/O read and write, and configuration read and write. Ex. 1011 at 4-5, Fig. 5, Ex. 2001 at 21; Ex. 2021 at ¶¶ 73, 114. Configuration transactions are required by the PCI standard in order to initialize the PCI bus to accept PCI transactions. Ex. 2021 at ¶ 95. Because configuration transactions are

not supported on TNet, TNet *cannot* communicate PCI bus transactions. Ex. 2015 at 9; Ex. 2021 ¶ 143. Even if the PCI addressing scheme could work over TNet, which it cannot, there is no way to configure the TNet devices to operate with PCI transactions. *Id.* at ¶¶ 141-46.

Nor does Horst disclose any mechanism for the TNet packets to communicate PCI bus commands, which would indicate a type of PCI standard bus transaction. Ex. 1011 at 4-5; Ex. 2001 at 21; Ex. 2021 at ¶¶ 73, 76, 89, 111, 114, 119. Instead, the only commands Horst discloses for TNet relate to link-level flow control, initialization, and error signaling. Ex. 1011 at 3, Table 1; Ex. 2021 ¶ 37. In summary, TNet does not support CPU-side PCI bus transactions.

Further, the TNet packet, as disclosed in Horst, does not disclose PCI standard addresses. Ex. 1011 at Fig. 5. First, TNet discloses that it does not support memory- mapped I/O (Ex. 1011 at 6), which is the addressing scheme used in PCI memory read-and-write transactions. Ex. 2021 ¶ 113. Second, TNet does not support any PCI standard addresses. *Id*. TNet addresses are virtual and PCI addresses are physical, which are two vastly different addressing schemes. *Id.* ¶ 38-41, 113. Only after a TNet packet is routed to a remote TNet node connected to a PCI bus does the intelligent ASIC in the PCI bus node generate a PCI bus

transaction based on the TNet packet information and other information contained in the ASIC. *Id.* ¶ 103, 105, 116.

III. CONSTRUCTION OF AND SUPPORT FOR THE INSTITUTED CLAIMS

A. The Standard for Claim Construction in These Proceedings

The claims are to be given their broadest reasonable interpretation in light of the specification. *See* 37 C.F.R. § 42.100(b).

B. Patent Owner's Proposed Claim Constructions

Patent Owner's proposed claim constructions for selected claim terms and clauses are provided below. Any claim terms or clauses not expressly construed herein should be given their plain and ordinary meaning as understood by one skilled in the art.

1. Peripheral Component Interconnect (PCI) bus transaction

The proper construction for "Peripheral Component Interconnect (PCI) bus transaction" and "PCI bus transaction" is the one the Board adopted on institution—"Peripheral Component Interconnect (PCI) industry standard bus transaction." Any other reading of the term would render the phrase "PCI" meaningless.

The POSA would understand that a "PCI bus transaction" refers to a **PCI standard bus transaction** and not just any type of bus transaction. The PCI industry standard was well known at the time of the invention, and the POSA

would not confuse the PCI transaction for any other transaction. Ex. 2021 ¶ 119. The PCI version 2.1 local standard bus specification is dated June 1, 1995, three years prior to the '873 patent priority date. Ex. 2001 at 1.

Therefore, the Board's construction is the proper construction under BRI.

1. Encoded

Claim 54 recites "an encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction." Encoding can apply to different types of operations in the field of computing; however, two conditions are required for all encoding operations involving numerical values: (i) reversibility and (ii) uniqueness of coded values. Ex. 2024 at 4 (1994 IBM technical dictionary at 111). According to Petitioner's expert, Mr. Young, reversibility in the '873 patent requires that "none of the data actually changes. It just is the order its put in." Ex. 2022 at 11:2-5. The reason for this is so that "by knowing the order, …, you know how to then tease the bits back out on the other end and put them in the right order on the other side." *Id.* at 10:20-25. For the decoder to put the bits in the right order, the order in which the bits are encoded must be shared between the encoder and decoder. *Id.* at 11:10-18. Uniqueness is a condition of reversibility and requires that every coded value must relate to only one unencoded value. Ex. 2021 at XX. This is the reason standards are critical—they allow devices to communicate because they define the order of the bits and what those bits mean.

While encoding can typically have a variety of meanings within this framework, the '873 patent describes a very limited set of operations that could be considered encoding. In fact, Petitioner's expert, Mr. Young opined that the '873 patent only described one form of what the POSA would consider encoding—ordering bits onto multiple serial channels. Ex. 2022 at 8:11-10:7. While the '873 patent does indeed disclose this type of encoding, it additionally discloses two other operations that the POSA would consider encoding. The '873 patent discloses turning PCI bus signals into bits for serial transmission, which the POSA would recognize as a type of encoding. Ex. 1001 at 5:33-36; Ex. 2021 ¶ 122. The '873 patent also discloses grouping the bits into specified size blocks of, for example, 9 bits with a 1-bit clock synchronization to be sent on LVDS multi serial bit channels. Ex. 1001 at 17:54-60, 21:61-65; Ex. 2021 ¶ 123.

Encoding in the form of turning signals into bits is described in the specification as encoding the control signals into control bits. Ex. 1001 at 5:33-36. The POSA would recognize that this is a form of encoding. Ex. 2021 ¶ 122. The specification explains that "the control bits representing control signals are decoded back into PCI control signals" before being transmitted on the PCI bus. Ex. 1001 at 5:36-39.

Encoding in the form of grouping bits into specified size blocks of bits is described in the specification as grouping the bits into blocks for transmission on the serial lines. Ex. 1001 at 17:54-60, 21:65-67; figs 13 & 14. The POSA would recognize this as a form of encoding. Ex. 2021 ¶ 123. Figures 13 & 14 from the '873 specification show turning the bits into 8 bit blocks of data for transmission on the LVDS serial channels with an additional bit for clock synchronization.

Encoding in the form of ordering the bits onto the serial channels is described in the specification in the description of the encoders that "format the PCI address/data bits to a form more suitable for parallel to serial conversion" before being sent over the serial line. Ex. 1001 at 16:55-58; 17:11-13. Ordering the bits onto the serial channels is also shown in Figures 13 & 14, where the address and data bits of a PCI bus transaction are ordered into 4 and 8 bit strings. Ex. 1001 at FIGs. 13 & 14. The decoders on the receiving end are then able to reverse the ordering for transmission over the PCI bus. *Id.* at 16:58-60.

Therefore, the BRI of "encode" in light of the specification is "a reversible operation that (1) turns a signal into bits, (2) groups bits into a specified size block, or (3) orders bits onto one or more serial transmission lines."

IV. THE INSTITUTED CLAIMS ARE PATENTABLE OVER THE INSTITUTED GROUNDS

Horst does not disclose key limitations of claims 54 and 56 – 61. These claims are very specific on what is being communicated, how the communication happens, and where the communication happens. And these specifics distinguish the claimed invention from Horst. Combining these references with the secondary references does not cure the problem. Notably, these secondary references are not even offered to teach the limitations missing from Horst.

A. The Instituted Claims Are Patentable Under 35 U.S.C. §§ 102 and 103

1. Claims 54, 57, 60, and 61 are not rendered obvious over Horst and the LVDS Owner's Manual; (Ground 1)

Claim 54 requires a peripheral bridge directly coupled to an LVDS channel that communicates an "encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction." Ex. 1001 at claim 54. The claim also requires that the peripheral bridge be "directly coupled" to the microprocessor unit. Ex. 1001 at claim 54. Horst does not teach this architecture. Instead, as discussed further below, Horst uses only TNet transactions on the CPU-side of the bus. Ex. 1011 at 2, 8, Figs. 2, 9. Horst creates a parallel (not serial) PCI bus transaction on the peripheral side of the computer system using specialized hardware and software. Horst never serializes any PCI bus transaction, even the peripheral-side

PCI bus transaction. The only PCI bus transaction shown in Horst is parallel and is on the peripheral side. Thus, Horst fails to teach this claim limitation. And no other reference is put forward to teach this portion of the claimed invention.

Claim 61 further requires that the encoded serial bit stream comprises "encoded PCI address and data bits." Again, Horst does not teach this architecture. Horst does use a serial communication method between the CPU side of the system and the peripheral devices, but it does not communicate PCI address and data bits over that link. Ex. 2021 at ¶¶ 140-146. Horst's uses of the TNet protocol means that the PCI address bits are not transmitted over the TNet links. Horst defines the packet format in figure 5. Ex. 1011 at Figure 5. A PCI address is not included. ***Id.***

At best, Horst transmits information that specialized hardware and software on the peripheral side can use to later generate a PCI bus transaction. *Id*. at ¶¶ 140-146. That PCI bus transaction is then communicated in *parallel* form. ***Id.*** ¶¶ 63, 119, 136, 144. Accordingly, Horst fails to teach this limitation. And no other reference is put forward to teach this portion of the claimed invention.

Consequently, neither Horst nor any combination with Horst teaches the system claimed in claim 54 or any of its dependent claims, including claims 56-61. Horst teaches a different architecture and never serializes an actual PCI bus transaction.

Finally, as Dr. Lindenstruth confirms, it would not be obvious to communicate using PCI over the TNet link. Ex. 2021 ¶¶ 147-156. First, Horst teaches away from using PCI transactions over the TNet link. *Id.* ¶¶ 149-151. Horst explicitly considered standard busses like PCI and discarded them in favor of a completely new proprietary network. *Id.* ¶ 149. Second, communicating PCI over TNet would render the system inoperable because the memory structure of TNet is incompatible with PCI transactions. *Id.* ¶ 150. Third, even if the system was not rendered inoperable, communicating PCI bus transactions over TNet would add unnecessary complexity and expense to a functioning system. *Id.* ¶ 155.

The subsections below address these points in more detail.

a. Horst does not disclose a peripheral bridge directly coupled to the processing unit and the LVDS channel that communicates "an encoded serial bit stream of PCI bus transaction"[2]

Claim 54 recites, in part:

> "a low voltage differential signal (LVDS) channel … for communicating an encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction;"

[2] ACQIS does not dispute the presence of a low voltage differential signal (LVDS) channel, but does dispute the Board's construction and does not acquiesce to it. The construction of a low voltage differential signal channel has not been argued and its presence is not relevant to ACQIS's validity arguments.

and

> "a peripheral bridge directly coupled to the processing unit, the peripheral bridge comprising an interface controller directly coupled to the LVDS channel."

Ex. 1001, claim 54. This claim requires that the peripheral bridge be directly coupled to both the processing unit and the LVDS channel and that it communicate an encoded serial bit stream of PCI bus transaction. Therefore, the PCI bus transaction must be a serial bit stream and that serial bit stream must be encoded. Nothing in Horst discloses a peripheral bridge that communicates an encoded serial bit stream of *PCI* bus transaction.

(1) Horst's "peripheral bridge" is directly coupled to the microprocessor

Petitioner asserts that Horst discloses a peripheral bridge in the form of a TNet processor interface, which is directly coupled to the microprocessor. Paper No. 2 at 32. The excerpted and annotated figures at Figure D show the relationship of the TNet processor interface in relation to the overall system architecture. Horst's figure 9 (on the left) shows the high level architecture of a typical system using TNet, including the processor interface. Horst's figure 7 shows a detailed view of that processor interface.

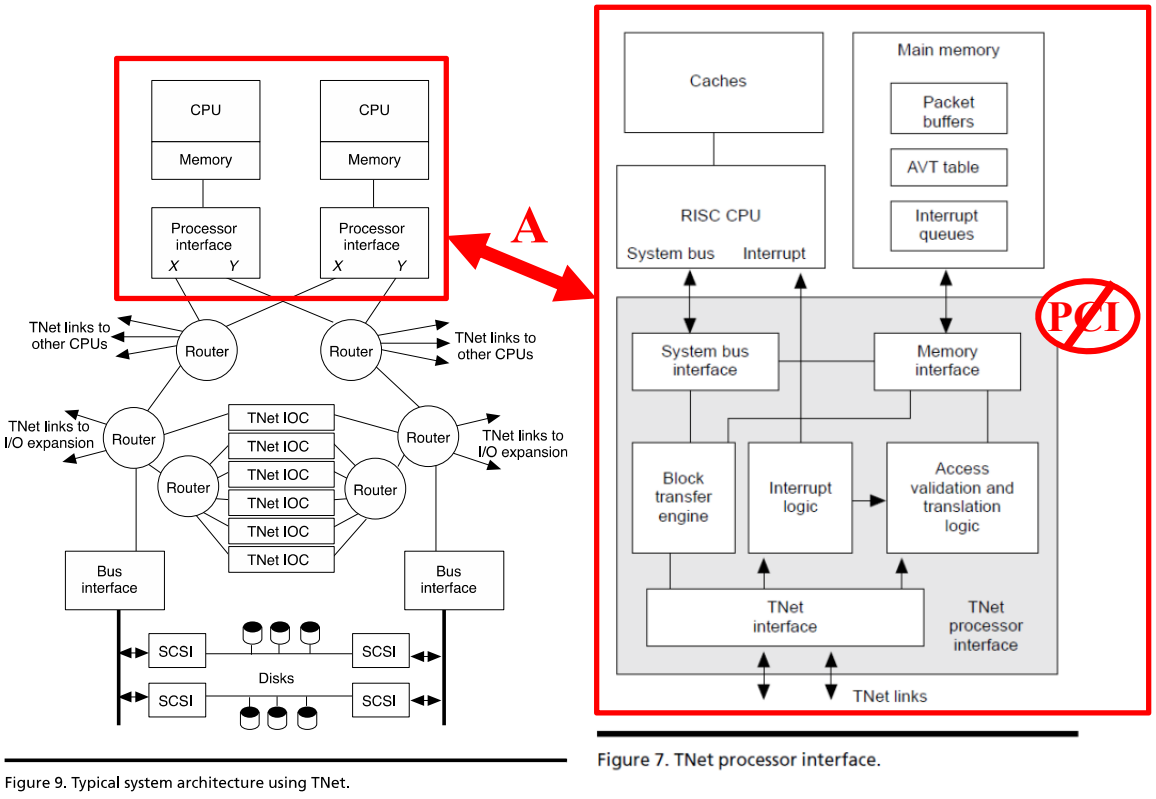


Figure 9. Typical system architecture using TNet.

Figure 7. TNet processor interface.

Figure D: TNet Processor Interface in Relation to the Typical TNet Architecture

As shown by the arrow labelled "A," the TNet processor interface is directly coupled to the CPU-memory bus. There are no other CPU-side interfaces or busses described in Horst that could meet this limitation. Ex. 2021 at ¶¶ 103, 109, 110.

(2) Horst's TNet links are the only LVDS serial channels coupled to the "peripheral bridge"

The TNet processor interface is coupled to the TNet processors, the TNet memory, and the TNet links. Ex. 1011 at 6, Fig. 7. The TNet links are low voltage

byte-serial channels. Ex. 1011 at 3, Fig. 3. No other serial channels are disclosed connected to the TNet processor interface.

(3) Horst's "peripheral bridge" does not communicate an encoded serial bit stream of PCI bus transaction over a LVDS channel

The TNet processor interface does not communicate an encoded serial bit stream of PCI bus transactions as required by claim 24. BRI of "encode" is "a reversible operation that (1) turns a signal into bits, (2) groups bits into a specified size block, or (3) orders bits onto one or more serial transmission lines."

The only form of encoding a serial bit stream disclosed in Horst is the 8b/9b encoding of the symbols, which the Board recognized. Paper No. 14 at 11; Ex. 1011 at 3. 8b/9b encoding is only related to evening out voltage signals on the serial channel. Ex. 2021 at ¶ 115. Dr. Lindenstruth confirms that 8b/9b encoding is incapable of translating a parallel PCI transaction into a TNet transaction. *Id*. In fact, Horst states the 8b/9b encoding is applied to TNet transactions, not to PCI transactions. Ex. 1011 at 3. Therefore, simply because the TNet processor interface encodes the TNet transactions, this does not mean TNet is communicating encoded serial bit stream of PCI bus transaction.

To find that the peripheral bridge communicates an encoded serial bit stream of PCI bus transaction in Horst, the TNet transaction must be, or include, a serial PCI transaction. As shown in Figure D above, nothing suggests that the TNet

processor interface communicates any part of a PCI bus transaction, encoded or not. Ex. 1011 at 6 Fig. 7; Ex. 2021 at ¶¶ 103-104, 144.

TNet Cannot Transmit Serial PCI Bus Transactions

Dr. Lindenstruth confirms that Horst teaches a POSA that TNet transactions cannot be serial PCI standard transactions. Ex. 2021 at ¶¶ 140-147. The PCI standard sets out specific requirements for PCI standard addresses. Critically, all PCI standard addresses are physical addresses in one of three address spaces. Ex. 2001 at 42; Ex. 2021 ¶ 113. The only addresses disclosed in Horst that are communicated over TNet are virtual TNet addresses. Ex. 1009 at 5, Fig. 5; Ex. 2021 ¶ 113. The format of a TNet packet is shown in figure 5 of Horst, excerpted and annotated below as Figure E.

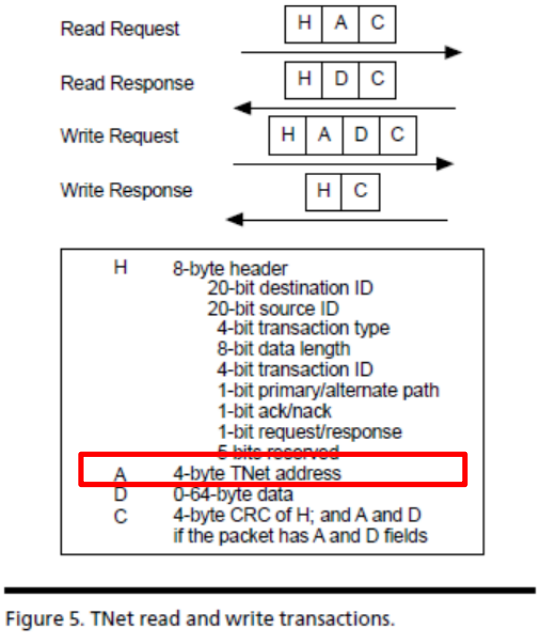


Figure E: The TNet Packet

Horst states that TNet address are virtual addresses, not physical addresses. Ex. 1009 at 2; Ex. 2021 at ¶ 108, 113. This means that TNet addresses cannot be PCI standard addresses because physical addresses cannot be virtual addresses. Ex. 2021 at ¶¶ 138-139; Ex. 2001 at 42. Upon configuration, a PCI bus master must map the PCI address of each PCI device into a specific physical address location in the processor's memory or I/O address space. *Id.* at ¶ 67, 145. This is a flat addressing scheme, which is an addressing scheme for a single processor node address space. *Id.* ¶ 138.

In addition, TNet transactions cannot be serialized PCI transactions because sending PCI transactions across TNet would render the TNet system non-functional. The PCI standard uses a flat address space. Ex. 2015 at 12; Ex. 2016 at 6; Ex. 2017 at 3; Ex. 2021 ¶ 152. This means that every device on a PCI bus is assigned a unique address in the PCI bus host processor's address space. *Id.* Further, in a PCI standard bus, every device decodes the address of every PCI transaction. Ex. 2001 at 42. TNet addresses do not use a flat address space because TNet interconnects multiple processor nodes, each with its own 32-bit address space. Ex. 1011 at 5. Therefore, flat addressing scheme used by PCI would not work on TNet. *Id.*

Dr. Lindenstruth confirms that PCI standard transactions are incapable of functioning in the TNet multiple processor node system. Ex. 2021 ¶ 154. Because each TNet node device has a different address space with the same address range each PCI transaction would address every processor node on TNet resulting in a massive number of unintended transactions. Ex. 2016 at 6; Ex. 2017 at 3; Ex. 2020 at 3. This makes sense because TNet specifically replaced the PCI local bus with the TNet network to allow it to interface multiple processor nodes. Ex. 1011 at 1-2, Figs. 1-2.

Further, TNet does not support the PCI standard bus transactions on the reverse side. The PCI standard bus transactions include memory, I/O, and configuration read and write transactions, each of which use a different address space. Ex. 2001 at 41; Ex. 2021 ¶ 73. TNet only discloses generic read, write, and unposted write transactions. Ex. 1011 at 5, Fig. 5. Nor does TNet disclose any means of transmitting PCI bus commands that control the type of PCI bus transaction. Ex. 2021 at 111, 114. Without the ability to transmit PCI standard bus commands, a POSA would realize TNet would not function if it used PCI bus transactions. *Id.* ¶ 89.

Additionally, the PCI standard requires PCI configuration transaction be supported. Ex. 2001 at 42. The PCI host processor uses PCI configuration

commands to initialize the PCI bus on startup. Ex. 2001 at 42. If TNet supported PCI transactions, at startup each TNet processor would attempt to initialize the TNet network and map the PCI address of each device on the TNet network into a specific physical address location in the host processor's memory or I/O address space. Ex. 2016 at 6; Ex. 2017 at 3; Ex. 2020 at 3. This means that each TNet processor node would attempt to map every other TNet node into its address space, which would result in address collisions. Ex. 2015 at 12; Ex. 2016 at 6; Ex. 2017 at 3; Ex. 2020 at 3; Ex. 2021 ¶ 79.

As a result, it makes sense that Horst discarded the PCI standard in favor of TNet when designing a multiple processor node system for memory-to-memory transactions.

Horst Removed PCI when Designing TNet

Figure F (excerpted and annotated figures 1 & 2 from Horst), below, illustrates that Horst designed the TNet system to specifically remove the PCI bus from the CPU side to be able to communicate between CPUs. Horst Figure 1 (left) shows a traditional, PCI-based system. Horst Figure 2 (right) shows a typical TNet architecture. Horst had to replace the PCI bus in the traditional system because it is incapable of performing the CPU-to-CPU communication Horst desired. Ex. 2021 ¶ 103. Horst replaced the CPU-side PCI bus with the TNet link. Where in the traditional system (left) the PCI interface existed on the CPU side, Horst intentionally moved the PCI interface into the peripheral side in the TNet system (right), where it resides at the same level as all the other bus interfaces.

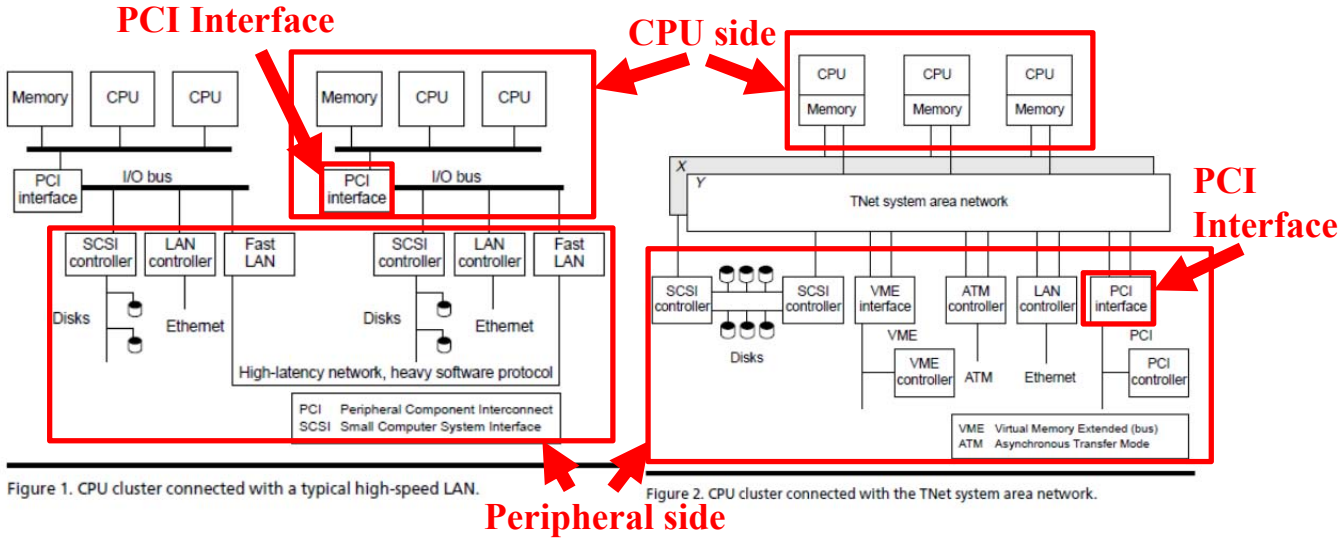


Figure F: Comparison of Traditional PCI-based Architecture and TNet Architecture

The TNet processor interface (shown in detail in Figure D above) is directly coupled to the CPU-memory bus and therefore does not start with a PCI bus transaction to translate. Additionally, the TNet processor interface translates the CPU-memory bus transaction into a TNet packet, which does not include PCI address bits. *Id.* ¶ 103, 105, 144. Instead, the TNet bus interface ASIC generates the parallel PCI bus transaction on the receiving end for transmission on the parallel PCI bus. *Id.* Therefore, the TNet processor interface, which is on the CPU side, does not communicate address bits of PCI bus transaction. *Id.* 140-146.

No CPU-Side PCI transaction

Horst discloses that there is never a CPU-side PCI transaction. Because the TNet processor interface adapter intakes only CPU-memory bus transactions on the system bus interface, there is never a PCI bus transaction on the CPU side in the first place. *Id.* at ¶ 103, 165. Instead, the TNet processor interface generates a TNet packet from a CPU-memory bus transaction for transmission on the TNet link. Figure F; *Id.* ¶ 103. This is highlighted by the "Access validation and translation logic" block that handles incoming and outgoing traffic. Ex. 2021 109-112.

Only PCI Transaction is a Remote Parallel PCI Transaction

Switching now to the peripheral side, the only PCI transaction in the TNet system occurs on the peripheral side of the TNet to PCI bus interface. Ex. 1011 at Fig. 2; Ex. 2021 ¶ 103, 105, 144. The excerpted and annotated figures at Figure G show the relationship of the TNet bus interface in relation to the overall system architecture with the arrow labelled "B." Horst's figure 9 (on the left) shows the high level architecture of a typical system using TNet. Horst's figure 8 shows a detailed view of the peripheral side of a TNet system.

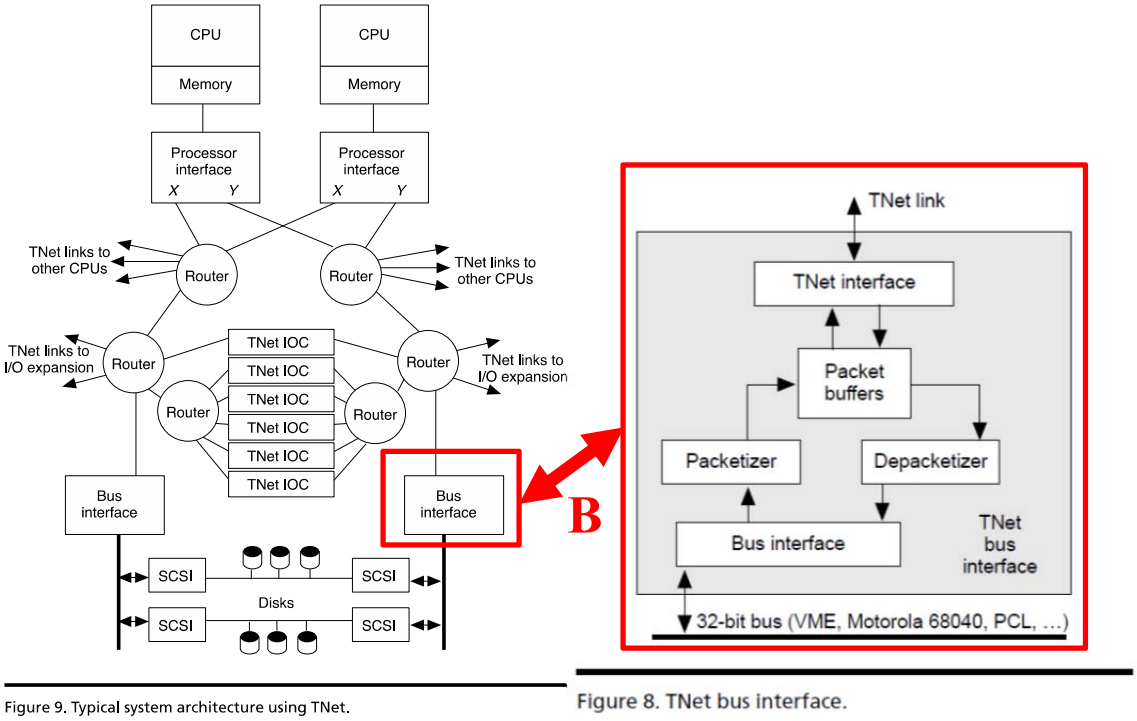


Figure 9. Typical system architecture using TNet.

Figure 8. TNet bus interface.

Figure G: TNet Bus Interface in Relation to the Typical TNet Architecture

The TNet bus interface receives the TNet packet over the TNet link and translates it into a bus transaction for whichever 32-bit bus it is coupled to (e.g., VME, PCI, or SCSI). The TNet bus interface uses an intelligent ASIC, which a POSA would understand generates the appropriate bus commands and addresses based on the particular bus configuration. Ex. 2021 at ¶ 103, 105, 139. As a result, the TNet transaction need not be a serialized PCI standard bus transaction for the intelligent TNet PCI interface ASIC to generate a PCI bus transaction on the peripheral side, parallel PCI bus. *Id.* ¶ 141. That the TNet bus interface is intelligent makes sense because TNet needs to interact with multiple processors and bus interfaces, each connecting to a different bus protocol with different devices. *Id*.

The system works similarly in reverse. In that case, the TNet bus interface receives a PCI transaction from the peripheral side PCI bus which is parallel, not serial. *Id.* ¶ 111. The intelligent ASIC in the TNet bus interface generates a TNet transaction for transmission over the TNet link. *Id.* ¶ 111-114.

Note that because Horst removed the CPU side PCI bus, extensive specialized hardware and software (the TNet bus interfaces) are required for TNet to function. *Id.* ¶ 116. Therefore, modified drivers are required for all peripheral devices. *Id.* As discussed above, a key advantage to the claims of the '873 patent

is the compatibility with legacy devices. *Id.* ¶¶ 75, 85-90; *Supra* at 3. TNet eliminates this advantage.

The presence of the TNet bus interface on the peripheral side of Horst does not demonstrate that the TNet link carries an encoded serial bit stream of PCI bus transaction as recited in claim 54. In fact, it shows the opposite. The TNet bus interface would not be necessary if the TNet link were capable of carrying an encoded serial bit stream of PCI bus transaction. Ex. 2021 ¶ 103-105, 116. Horst's TNet bus interface is required to convert the TNet information coming off of the TNet link from the CPU into the PCI protocol format, including by adding/appending the PCI address so that the data can be carried on the only PCI bus in the TNet system: the peripheral-side, parallel bus. *Id.* ¶ 144. For the information coming off of Horst's peripheral-side, parallel PCI bus into the TNet bus interface, the TNet bus interface is necessary to strip away the PCI information, such as the PCI address, and convert the data to the format that is supported by Horst's TNet link, which does not include the required information to qualify as a PCI bus transaction (e.g., a PCI address). *Id.* ¶ 111. Horst's TNet bus interface even requires special hardware—an ASIC—to accomplish this conversion because Horst's TNet link—the only serial bus in Horst—cannot carry an encoded serial bit stream of PCI bus transaction. *Id.*¶ 105, 117.

TNet does not use PCI for its entirely new and proprietary network on the CPU side. Ex. 2021 ¶ 102. The new network (TNet) described by Horst uses virtual addressing and address translation and purposefully avoids memory-mapped I/O and physical addressing (e.g., PCI) required for a PCI bus transaction. Ex. 1009 at 2, 6; Ex. 2021 ¶ 108. The TNet packets and transaction types do not allow for the PCI standard bus transactions, including memory read and write, I/O read and write, and configuration read and write. Ex. 1009 at 4-5, Fig. 5, Ex. 2001 at 21; Ex. 2021 at ¶ 105, 108. Nor does Horst disclose any mechanism for the TNet packets to communicate PCI bus commands indicating a type of PCI standard bus transaction. Ex. 1009 at 4-5; Ex. 2001 at 21; Ex. 2021 at ¶ 105. All these differences make clear that TNet simply does not match the claim limitations that require a peripheral bridge directly coupled to the processing unit and the LVDS channel that communicates "an encoded serial bit stream of PCI bus transaction." Ex. 2021 ¶ 117.

(4) "Encoded" cannot be used to read the PCI standard out of the claims

The BRI of "encoded" in claims 54 and 61 cannot be used to read the "PCI standard" out of the claims. Claims should be construed to give every term meaning. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005) (en banc).

Allowing the term "encoded" to read out "PCI standard" is therefore improper. The relevant claim language is below:

- Claim 54 "an encoded serial bit stream of Peripheral Component Interconnect (PCI) bus transaction;"
- Claim 61 "wherein the encoded serial bit stream of PCI bus transaction comprises encoded PCI address and data bits." (Ex. 1001 at cl. 54, 61.)

(a) Neither party's definition of encoding can transform a TNet transaction into a PCI transaction

TNet does not support PCI standard bus commands or disclose transmitting PCI standard bus commands over TNet. *Supra* at 26-36. As a result, PCI standard bus commands cannot be encoded into a TNet packet, no matter what definition of encoding is applied. Both Petitioner and Patent Owner agree that, at a minimum, encoding must be reversible. As Petitioner's expert confirmed, this requires that the information be present and that the recipient know how to reorder the information to recover the transaction. Ex. 2022 at 10:20-11:18. Since there is no disclosure of transmitting PCI bus commands on the TNet links, there is no way for the recipient to recover the PCI bus transactions from the TNet transaction. This means that the PCI bus transactions are not reversible and are therefore not encoded.

(b) The Board's properly applied BRI of encoding in Horst cannot transform a TNet transaction to a PCI bus transaction

Even if the PCI bus commands were present, Horst's disclosure of encoding cannot otherwise transform a PCI bus to a TNet transaction or vice versa. The Board properly applied the BRI of encoding in its analysis of Horst by pointing to the 8b/9b line encoding. Paper No. 14 at 11. This type of encoding meets the BRI of encoding as reversibly grouping bits into a specified size block—in this case 9-bit blocks. *Supra* at 21. However, Dr. Lindenstruth confirms that 8b/9b encoding cannot transform a PCI bus transaction address into a TNet transaction virtual address or vice versa. Ex. 2021 at ¶ 115. Instead, 8b/9b encoding uses an algorithm to balance the electrical load on TNet's differential signal channels. *Id.*

(c) None of the other BRIs of encoded can transform a TNet transaction into a PCI bus transaction

The remaining BRIs of encoding in the claims, transforming signals into bits or ordering bits onto multiple serial channels, are incapable of transforming TNet packets into PCI bus transactions. First, TNet does not use PCI bus transaction types or disclose transmitting PCI bus commands. Ex. 2021 at ¶¶ 140-141; *Supra* at 9, 15, 36. As a result, there are no PCI bus command signals to convert to bits for transmission over TNet or vice versa. Neither are there any PCI bus command bits for ordering onto serial channels.

Second, the BRI of encoding as transforming signals into bits or ordering bits onto multiple serial channels cannot transform a PCI address into a TNet virtual address or vice versa. Instead, virtual addresses require replacing the physical address with a different virtual address. Ex. 1011 at 2; Ex. 2021 at ¶¶ 39-40. The mapping between the virtual address and the physical address is different for each transaction and known only to the transmitting entity, which maintains a page file containing the address validation and translation table for each transaction. Ex. 1011 at 2, 5, 6 Fig. 7 (AVT table); Ex. 2012 at 4-11; Ex. 2013 at 3. As a result, the receiving entity has no way to generate the physical address from the virtual address. Ex. 2021 at ¶¶ 39-40, 138-139. Therefore, the virtual address is not reversible and is therefore not an encoded physical PCI address. *Id.* at ¶¶ 39-40, 138-139.

In fact, Horst takes advantage of this non-reversible aspect of virtual addressing for security purposes to prevent an unauthorized I/O device from writing directly to processor memory. Ex. 1011 at 6 ("When an inbound read-or-write request packet arrives at the processor interface, the interface must validate and translate the virtual TNet address by looking it up in the AVT table.") Horst discloses that each AVT entry includes the physical address translation, permission

bits" and the "identity of TNet nodes allowed to use the [AVT] entry." Ex. 1011 at 6.

As a result, TNet's virtual address cannot be encoded PCI bus transaction addresses. Dr. Lindenstruth confirms TNet virtual address are not encoded PCI addresses. Ex. 2021 at ¶¶ 138-139.

Petitioner's Arguments Do Not Account for the Claim Requirements

Petitioner and Mr. Young appear to argue that, since a PCI bus transaction occurs at some point in the system on the peripheral side, every part of that system is communicating or transmitting that PCI bus transaction. Paper No. 2 at 30-33. The presence of a PCI bus interface on the peripheral side does not mean that the TNet processor interface on the CPU side arrives as a PCI bus transaction. Ex. 2021 ¶ 136. In fact, the TNet processor interface cannot transmit a PCI bus transaction as recited in claim 54. *Supra* at 31. The presence of the PCI bus interface on the peripheral side means only that the peripheral side can handle parallel PCI bus transactions.

Further, as a practical matter, Mr. Young's conclusion is illogical. By the same logic employed by Mr. Young, the presence of other peripheral interfaces, like the VME interface, would mean that the TNet processor interface is also

carrying VME bus transactions—and all other types of transactions associated with any communication protocol shown as attached to the TNet network in Horst.

(5) Horst's "peripheral bridge" does not communicate address bits of PCI bus transaction in serial form

The TNet processor interface (Horst's "peripheral bridge") does not communicate address bits of PCI bus transaction in serial form. Ex. 2021 ¶ 144. As noted above, the TNet processor interface does not communicate address bits of PCI bus transaction at all. *Supra*.

Notably, the PCI bus transaction is not generated until the TNet packet is received by the TNet bus interface for a PCI bus on the peripheral side. *Id.* ¶ 103. Once received, the TNet bus interface ASIC generates the PCI bus transaction, *Id.* ¶ 136, which is the first time that the PCI bus transaction exists in the system. *Id.* At that time, the PCI bus transaction is communicated *in parallel form* over the PCI bus. *Id.* ¶¶ 63, 119, 136, 144.

Dr. Lindenstruth confirms that Horst does not disclose a "peripheral bridge" directly coupled to the processing unit and the LVDS channel that communicates "an encoded serial bit stream of PCI bus transaction." *Id.* ¶ 117. Accordingly, Horst does not disclose or render obvious each limitation of claim 54 and its dependent claims. Therefore this Board should confirm claims 54, 57, 60, and 61 over Ground 1.

2. Claims 56 and 59 are not rendered obvious over Horst, the LVDS Owner's Manual, and Pocrass; (Ground 2)

As described in detail above, claim 54 is not obvious over Horst in combination with the LVDS manual. Because Pocrass does not remedy the deficiencies of the combination and Petitioner does not argue that it does, claims 56 and 59, which depend from claim 54, are also not rendered obvious over the combination of Horst, the LVDS manual, and Pocrass.

3. Claim 58 is not rendered obvious over Horst, the LVDS Owner's Manual, and Pocrass; (Ground 3)

As described in detail above, claim 54 is not obvious over Horst in combination with the LVDS manual. Because Deters does not remedy the deficiencies of the combination, and Petitioner does not argue that it does, claim 58, which depends from claim 54, is also not rendered obvious over the combination of Horst, the LVDS manual, and Deters.

V. CONCLUSION

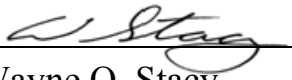
For the foregoing reasons, Patent Owner respectfully submits that Petitioner fails to establish by a preponderance of evidence that any of the Instituted Claims are unpatentable under any of 35 U.S.C. §§ 102 or 103. Accordingly, all of the Instituted Claims should be confirmed.

June 11, 2015

Respectfully submitted,
COOLEY LLP

COOLEY LLP
ATTN: Patent Group
1299 Pennsylvania Avenue, N.W
Suite 700
Washington, DC 20004-2400

Tel: (720) 566-4000
Fax: (202) 842-7899

By: 

Wayne O. Stacy
Reg. No. 45,125

CERTIFICATION OF SERVICE UNDER 37 C.F.R. § 42.6(e)

I, SARA J. BRADFORD, hereby certify that on 11th day of June the foregoing **PATENT OWNER'S RESPONSE** was served electronically via email on the following:

Brian Buroker
bburoker@gibsondunn.com

Blair Silver
bsilver@gibsondunn.com

c/o GIBSON, DUNN & CRUTCHER LLP
1050 Connecticut Avenue, N.W.
Washington, D.C. 20036-5306
202.955.8541(telephone)

JUNE 11, 2015

Respectfully submitted,
COOLEY LLP

COOLEY LLP
ATTN: Patent Group
1299 Pennsylvania Avenue, N.W
Suite 700
Washington, DC 20004-2400

By: /Sara J. Bradford/
Sara J. Bradford
Reg. No. 68,715

Tel: (720) 567-4009
Fax: (202) 842-7899